



C1-23-07

AF/2192  
f/s  
BRW

THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: M. Kawahito, et al

Date: January 22, 2007

Serial No.: 10/085,455

Filed: February 27, 2002

Docket No.: **JP920000420US1**

COMMISSIONER FOR PATENTS  
Board of Patent Appeals and Interferences  
Alexandria, VA 22313-1450

Sir:

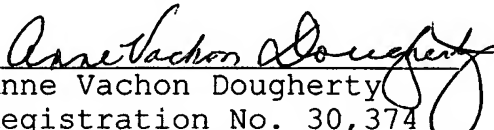
Transmitted herewith is an **Appeal Brief** for the above-identified Application along with a Petition for Extension of Time.

Applicants believe that the following fees are due at this time:

- Appeal Brief transmittal fee of \$500.

Authorization is hereby given to charge Deposit Account 50-0510 for those fees. Should any additional filing fee be required, the Commissioner is hereby authorized to charge payment of the fee associated with this communication to **Deposit Account No. 50-0510**.

Respectfully submitted,  
M. Kawahito, et al

  
Anne Vachon Dougherty  
Registration No. 30,374  
Tel. (914) 962-5910



Serial No. 10/085,455

I HEREBY CERTIFY THAT THIS CORRESPONDENCE IS  
BEING DEPOSITED IN EXPRESS MAIL AS  
EXPRESS MAIL: EB194546767US ON  
DATE OF DEPOSIT: January 22, 2007  
PERSON DEPOSITING: ANNE VACHON DOUGHERTY

Anne Vachon Dougherty 1/22/07  
Signature & Date

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In Re Application of : January 22, 2007  
M. Kawahito, et al : Group Art No.: 2192  
Serial No. 10/085,455 : Examiner: C. Pham  
Filed: February 27, 2002 : for IBM Corporation  
Anne Vachon Dougherty  
Title: PROGRAM OPTIMIZATION METHOD 3173 Cedar Road  
AND COMPILER USING THE Yorktown Hts, NY 10598  
PROGRAM OPTIMIZATION METHOD

Board of Patent Appeals and Interferences  
Alexandria, VA 22313-1450

**APPEAL BRIEF (37 CFR 41.37)**

Appellants hereby appeal to the Board of Patent  
Appeals and Interferences from the decision dated July 26,  
2006 of the Examiner finally rejecting Claims 1-16 in the  
above-identified patent application, and respectfully  
request that the Board of Patent Appeals and Interferences

01/24/2007 AWONDAF1 00000053 500510 10085455

01 FC:1402 500.00 DA

JP920000420US1

-1-

**Serial No. 10/085,455**

consider the arguments presented herein and reverse the Examiner's rejection.

### **I. REAL PARTY IN INTEREST**

The appeal is made on behalf of Assignee, International Business Machines Corporation and the inventors, Motohiro Kawahito, Hideaki Komatsu, John Craig Whaley, who are real parties in interest with respect to the subject patent application.

### **II. RELATED APPEALS AND INTERFERENCES**

There are no pending related appeals or interferences with respect to the subject patent application.

### **III. STATUS OF CLAIMS**

There are sixteen (16) claims pending in the subject patent application, numbered 1-16. No claims stand allowed. A complete copy of the claims involved in the appeal is attached hereto.

**Serial No. 10/085,455**

#### **IV. STATUS OF AMENDMENTS**

There are no unentered amendments filed after final rejection for the application.

#### **V. SUMMARY OF INVENTION**

The invention which is the subject of the remaining pending claims is a dynamic compiler, a dynamic compiling method, a computer, a storage medium and a support program for optimizing a program during compiling thereof. The present invention performs a dynamic analysis during execution to determine whether the execution speed of a program can be increased by fixing, in a specific state, a parameter for a predetermined command in the program; and then employs the results of the analysis to generate a path along which the parameter of the predetermined command is fixed in the specific state.

**Serial No. 10/085,455**

**Independent Claim 1**

As recited in independent Claim 1, the program optimization method for translating source code for a program written in a programming language into machine code and for dynamically optimizing the program comprises the steps of: performing a dynamic analysis during execution to determine whether the execution speed of said program can be increased by fixing, in a specific state, a parameter for a predetermined command in the program (page 20, lines 8-16) and employing results of the analysis for the dynamic generation, in said program, of a path along which the parameter of the predetermined command is fixed in the specific state (page 20, lines 19-22).

**Independent Claim 3**

As recited in Claim 3, a program optimization method for translating source code into machine code comprises the steps of executing a program to obtain statistical data for an appearance frequency of each available state in which a parameter of a predetermined command in said program may be set (page 24, line 15-page 25, line 20); and employing the obtained statistical data to dynamically generate a machine language program that includes, as the compiling results, a

**Serial No. 10/085,455**

path along which the parameter of the predetermined command is fixed in a specific state (page 27, lines 24-26).

**Independent Claim 5**

Claim 5 recites a program optimization method for translating the source code for a program written in an object-oriented programming language into machine code and for optimizing the program comprising the steps of detecting one command dynamically during execution, of the commands in said program, for which a method call destination can be identified, and for which the processing speed can be increased by identifying said method call destination (page 5, lines 16-19 and page 16, lines 7-8); and dynamically generating a path wherefor said method call destination for the detected command is limited in order to increase the processing speed of said command (page 5, lines 20-22 and page 16, lines 13-15).

**Independent Claim 6**

Claim 6 recites a program optimization method for translating into the source code for a program written in a programming language into machine code and for optimizing said program comprising the steps of detecting dynamically

**Serial No. 10/085,455**

during program execution one command, of the commands in said program, for which a variable can be limited to a predetermined constant value, and for which the processing speed can be increased by limiting the variable to the constant value (page 16, line 21-page 17, line 11 and page 18, lines 19-23 and page 38, lines 17-22); and generating a path along which said constant value for said variable of said detected command is fixed (page 18, lines 21-23 and page 40, lines 1-5).

**Independent Claim 7**

Claim 7 recites a dynamic compiler for translating into machine code the source code for a program written in a programming language, and for optimizing the resultant program comprising an impact analysis unit for performing an analysis to dynamically determine during execution how much the execution speed of said program can be increased by fixing, in a specific state, a parameter of a predetermined command in said program (12 of Figure 1 and page 14, lines 5-10); and a specialization unit for employing the analysis results obtained by said impact analysis unit to generate, in said program, a specialized path along which said

**Serial No. 10/085,455**

parameter of said predetermined command is fixed in said specific state (13-14 of Figure 1 and page 14, lines 15-27).

**Independent Claim 10**

Claim 10 recites a computer (page 7, lines 9-25) comprising an input device for receiving source code for a program; a dynamic compiler (10 of Fig. 1) for translating said source code to compile said program and for converting said compiled program into machine language code; and a processor for executing said machine language code, wherein the dynamic compiler includes means (impact analysis unit 12 of Fig. 1 and page 14, lines 5-10) for performing a dynamic analysis to determine during execution whether the execution speed of the program can be improved by fixing in a specific state a parameter of a predetermined command in said program, and means (data specialization selector 13 and specialization and compiling unit 14 of Figure 1 and page 13, lines 8-18) for generating in said program, based on the analysis results, a path along which the parameter of the predetermined command is fixed in the specific state and for compiling the program, and wherein said compiler outputs, as the compiled results, said machine language code that



**Serial No. 10/085,455**

includes said path along which the state of said parameter is fixed.

**Independent Claim 11**

Claim 11 recites a computer comprising an input device, for receiving source code for a program; a dynamic compiler (10 of Fig. 1), for translating said source code to compile said program and for converting said compiled program into machine language code; and processor, for executing said machine language code, wherein said dynamic compiler includes means (impact analysis unit 12 of Figure 1 and page 24, line 15-page 25, line 20) for obtaining statistical data for the appearance frequency of each available state wherein a parameter for a predetermined command in said program may be set when said program is executed, and for employing the statistical data to determine a state in which the parameter of the predetermined command is to be fixed, and means (data specialization selector 13 of Fig. 1 and page 20, lines 9-22) for generating a specialized path along which said parameter of said predetermined command is fixed in said determined state, and for compiling said program (specialization and compiling unit 14 of Fig. 1 and page 20, lines 9-22), and wherein said compiler outputs, as the

**Serial No. 10/085,455**

compiled results, said program as said machine language code that includes said specialized path.

**Independent Claim 13**

Claim 13 recites a support program (page 9, lines 1-10) for controlling a computer to support generation of a program which permits said computer to perform a function (Fig. 4 and page 22, line 24-page 5, line 3) for performing a dynamic analysis to determine during execution whether the execution speed of the program can be increased by fixing a parameter of a predetermined command of the computer program in a specific state; and a function for generating in the program, based on the analysis results, a path along which the parameter of the predetermined command is fixed in the specific state (Fig. 5, page 23, lines 4-20).

**Independent Claim 14**

Claim 14 recites a support program (page 9, lines 11-19) for controlling a computer to support generation of a program, which permits said computer to perform a function (Fig. 6, page 23, line 21-page 24, line 2) for executing said program and obtaining statistical data for the appearance frequency of each available state wherein the

**Serial No. 10/085,455**

parameter of said predetermined command of the program may be set; and a function for generating in the program, based on the statistical data, a path along which the parameter of said predetermined command is fixed in the specific state (page 24, lines 6-8).

**Independent Claim 15**

Claim 15 recites a storage medium (page 9, lines 20-22) on which input means of a computer stores a computer-readable support program (page 9, lines 1-3), for controlling said computer to support generation of a program, that permits the computer to perform a function for performing a dynamic analysis to determine during execution whether the execution speed of said program can be increased by fixing a parameter of a predetermined command of said computer program in a specific state (page 9, lines 4-7); and a function for generating in the program, based on the analysis results, a path along which the parameter of the predetermined command is fixed in the specific state (page 9, lines 7-10).

**Serial No. 10/085,455**

**Independent Claim 16**

Claim 16 recites a storage medium (page 9, lines 20-22) on which input means of a computer stores a computer-readable support program (page 9, lines 11-13), for controlling the computer to support generation of a program, that permits the computer to perform a function (page 9, lines 13-16) for executing the program and obtaining statistical data for the appearance frequency of each available state wherein the parameter of said predetermined command of the program may be set; and a function for generating in the program, based on the statistical data, a path along which the parameter of the predetermined command is fixed in the specific state (page 9, lines 16-19).

**VI. GROUND OF REJECTION TO BE REVIEWED**

The grounds of rejection in the Final Office Action included the following:

Claims 1-4 and 6-16 are rejected under 35 USC §102 as anticipated by Linden; and

Claim 5 is rejected under 35 USC §103 as unpatentable over Linden in view of Shaylor.

**VII. ARGUMENT**

**I. Rejections under 35 USC §102 as anticipated by Linden**

**Claims 1, 7, 10, 13 and 15**

The invention is a dynamic compiler (Claim 7), compiling method (Claim 1), computer (Claim 10), storage medium (Claim 15) and support program (Claim 13) for optimizing a program during compiling. The compiling invention performs a dynamic analysis during execution to determine whether the execution speed of the program can be increased by fixing, in a specific state, a parameter for a predetermined command in the program and then employs the results of the analysis to generate a path along which the parameter of the predetermined command is fixed in the specific state. Since the invention generates the path along which the parameter will be fixed in the specific state, it provides for dynamic specialization in the instances when the call method cannot be specified at a location whereat the method call is to be issued.

**Serial No. 10/085,455**

The Linden patent publication is directed to a dynamic compiler and method in which code is translated to run on a target machine that is different from the machine for which the code was developed. Under the Linden teachings, the intent and purpose of original instructions in the code are evaluated and new instructions are generated to arrive at the same result on the target machine. Linden says, at paragraph [[0041]], that the method provides "an interpolation from the source instructions to the equivalent results to be achieved by the target processor, independent of the operations specified by the source instructions". Linden creates new instructions for the target device "independent of operations specified by the source instructions" whereas the present invention uses the "predetermined command in said program" but dynamically generates a path along which a parameter for the command is fixed in a specific state.

The language of Claims 1, 7, 10, 13 and 15 expressly recite determining whether the execution speed can be increased by fixing a parameter for a predetermined command in a specific state and dynamically generating a path along which that parameter is fixed in that state. Linden does not look at a parameter of a predetermined command; rather,

**Serial No. 10/085,455**

Linden looks are intent and purpose of instructions. Further, Linden does not dynamically create a path along which a parameter of a predetermined command is fixed in a specific state. Linden teaches translating an instruction sequence into another sequence where the result is always a constant (paragraph [[0038]]), but that is not the same as or suggestive of dynamically creating a path along which a parameter of a predetermined command is fixed in a specific state. Appellants reiterate that Linden is not maintaining a predetermined command, is not determining a parameter for a predetermined command, and is not dynamically creating a path along which a parameter of a predetermined command is fixed in a state.

The Examiner has argued that "Linden clearly discloses optimizing the source program by generating an (sic) new instruction for the source instruction where the result is always a constant, for example by eliminating certain operations". Appellants respectfully assert that eliminating operations is not claimed. Appellants further assert that eliminating operations is not the same as the claimed step of determining whether execution speed can be increased by fixing a parameter for a command of the program in a specific state. Finally, Appellants assert that

**Serial No. 10/085,455**

Linden's step for making a result a constant does not anticipate fixing a variable parameter.

Appellants respectfully submit that the Examiner has not cited any teaching in Linden which anticipates the claim language of "performing a dynamic analysis...". The Examiner has listed terms including "dynamically cross-compiling, execution speed, execution time, overhead... dynamic recompilation... decoded instruction, instruction sequence, result, constant, Register3=0" on page 5 of the Final Office Action. Similarly, with respect to the claim language of "employing results..." the Examiner has listed terms (e.g., "instruction sequence, result, constant, Register=3, optimization step 44, optimized instruction stream" on page 5-6 of the Final Office Action), but the Examiner does not explain how Linden's use of those terms anticipates the claim language. Appellants contend that the Examiner's duty to apply teachings of the cited art to the claims in order to establish anticipation, and not to simply list terms from the reference.

Moreover, Appellants respectfully assert that the optimization stage and optimization steps disclosed by Linden do not anticipate the invention as claimed. In the optimization stage detailed by Linden in paragraphs [0040]



**Serial No. 10/085,455**

through [0047], there is no teaching or suggestion of performing a dynamic analysis during execution to determine whether execution speed of the program can be increased by fixing a parameter for a predetermined command in the program in a specific state. Linden states in paragraph [0041] that the output of the optimization stage is "an instruction stream that contains information on the equivalent flow of information to be handled by the target processor to emulate the intended results of the source instructions". Linden does not state that any parameters be fixed. Linden teaches storing fetched values [0042], allocating registers [0043], mapping registers [0044], overriding instructions [0046], and interleaving instructions [0047]. Linden does not, however, teach or suggest determining whether execution speed of a program can be increased by fixing a parameter for a predetermined command in the program and does not dynamically generate a path along which a parameter is fixed.

It is well established under U. S. Patent Law that, anticipation under 35 USC §102 is established only when a single prior art reference discloses each and every element of a claimed invention. See: In re Schreiber, 128 F. 3d 1473, 1477, 44 USPQ2d 1429, 1431 (Fed. Cir. 1997); In re

**Serial No. 10/085,455**

Paulsen, 30 F. 3d 1475, 1478-1479, 31 USPQ2d 1671, 1673 (Fed. Cir. 1994); In re Spada, 911 F. 2d 705, 708, 15 USPQ2d 1655, 1657 (Fed. Cir. 1990) and RCA Corp. v. Applied Digital Data Sys. Inc., 730 F. 2d 1440, 1444, 221 USPQ 385, 388 (Fed. Cir. 1984). Since the Examiner has not established that the Linden patent teaches steps or means for translating and optimizing a program comprising steps of performing a dynamic analysis during execution to determine whether the execution speed of said program can be increased by fixing, in a specific state, a parameter for a predetermined command in said program; or employing results of an analysis for the generation, in said program being compiled, of a path along which said parameter of said predetermined command is fixed in said specific state, it cannot be maintained that Linden anticipates the invention as set forth in the independent claims, Claims 1, 7, 10, 13 and 15, or the claims which depend therefrom and add further limitations thereto. Since the Linden patent publication does not teach the means and steps as claimed, it cannot be maintained that Linden anticipates the invention.

**Serial No. 10/085,455**

**Claims 2, 3-4, 8, 11-12, 14 and 16**

The claims further recite a dynamic compiler (Claims 8-9) dynamic compiling method (Claims 2 and 3-4), computer (Claim 11-12), storage medium (Claim 16) and support program (Claim 14) for translating source code to compile a program and for optimizing the program during compiling including steps and means for obtaining statistical data for the appearance frequency of each available state and for employing the obtained statistical data to dynamically generate the path

Appellants rely on the arguments set forth above with regard to the teachings of the Linden patent publication as they relate to the language of Claims 1, 7, 10, 13 and 15. Appellants further submit that the Examiner has not cited any teaching in Linden which anticipates the claim language of "obtaining statistical data for the appearance frequency of each available state" as is expressly claimed in Claims 2, 3-4, 8, 11-12, 14 and 16. In rejecting those claims, the Examiner lists terms used in the Linden reference (e.g., "instruction sequence, result, constant, Register=3, optimization step 44, optimized instruction stream" on page 6 of the Final Office Action), but the Examiner does not explain how Linden's use of those terms anticipates the

**Serial No. 10/085,455**

claim language. Appellants reiterate the contention that the Examiner has a duty to apply teachings of the cited art to the claims in order to establish anticipation, and not to simply list terms from the reference.

Moreover, Appellants respectfully assert that the optimization stage and optimization steps disclosed by Linden do not anticipate the invention as claimed. In the optimization stage detailed by Linden in paragraphs [0040] through [0047], there is no teaching or suggestion of obtaining statistical data for the appearance frequency of each available state wherein the parameter of a predetermined command may be set or of employing the obtained statistical data to dynamically generate the path (Claim 2). Similarly, there is no teaching or suggestion of a data specialization selector as set forth in Claims 8 and 9. Linden describes the output of its optimization stage and teaches operations of the optimization stage, such as providing an instruction stream that stores fetched values [0042], allocates registers [0043], maps registers [0044], overrides instructions [0046], and interleaves instructions [0047]. Linden does not, however, teach or suggest steps or means for obtaining statistical data for the appearance frequency of each available state and for employing the

**Serial No. 10/085,455**

obtained statistical data for determining and fixing a parameter.

Accordingly, Appellants conclude that the Examiner has established anticipation under 35 USC §102 since Linden does not disclose each and every element of a claimed invention as set forth in Claims 2, 3-4, 8, 11-12, 14 and 16. See: In re Schreiber, 128 F. 3d 1473, 1477, 44 USPQ2d 1429, 1431 (Fed. Cir. 1997); In re Paulsen, 30 F. 3d 1475, 1478-1479, 31 USPQ2d 1671, 1673 (Fed. Cir. 1994); In re Spada, 911 F. 2d 705, 708, 15 USPQ2d 1655, 1657 (Fed. Cir. 1990) and RCA Corp. v. Applied Digital Data Sys. Inc., 730 F. 2d 1440, 1444, 221 USPQ 385, 388 (Fed. Cir. 1984).

**Claim 9**

Claim 9 depends from Claim 8 and further recites wherein, in accordance with the state of said program at execution, the specialization unit generates a branching process for selectively performing a specialized path and an unspecialized path; and wherein, while taking into account a delay due to the insertion of the branching process, the data specialization selector determines a state in which the parameter of the predetermined command is fixed.

**Serial No. 10/085,455**

Appellants again rely on the arguments set forth above. Appellants further assert that the instructions listed by the Examiner on page 10 of the Final Office Action, including "*optimized instruction flow stream, optimization rules, pipeline delay*", do not provide teachings which anticipate a specialization unit which generates a branching process for selectively performing both a specialized path and an unspecialized path, or a data specialization selector for determining a state in which the parameter of a predetermined command is fixed.

Appellants reiterate that the Examiner has not provided teachings which anticipate the invention as claimed and that the Linden patent publication does not teach or suggest the invention as claimed in Claim 9.

**Claim 6**

Claim 6 recites a program optimization method for translating into machine code, the source code for a program written in a programming language, and for optimizing the program comprising the steps of dynamically detecting one command during program execution for which a variable can be limited to a predetermined constant value, and for which the processing speed can be increased by limiting the variable

**Serial No. 10/085,455**

to a constant value; and generating a path along which the constant value for the variable of the detected command is fixed.

Appellants again rely on the arguments set forth above regarding the teachings of the present invention and those of Linden. Appellants further assert that Linden does not anticipate the language of Claim 6. The Examiner has again cited a list of instructions in rejecting the claim language. The Examiner cites paragraph [0038] against Claim 6 because paragraph [0038] includes the term "constant". What Linden teaches is to "translate the instruction sequence into another sequence where the result is always a constant; e.g., Register 3=Register 1 XOR Register 1 (translate to Register 3=0)". Linden does not teach dynamically detecting a command for which a variable can be limited to a predetermined constant value. Nor does Linden teach generating a path along which the constant value for the variable of the detected command is fixed. Linden teaches translating an instruction sequence into another sequence where the results is always a constant (paragraph [[0038]]), but that is not the same as or suggestive of limiting a *variable* of an *existing command* of the program to a constant. Linden is not limiting a variable for a command

**Serial No. 10/085,455**

nor is Linden generating a path to ensure that the variable is fixed. Rather, Linden is translating an instruction sequence to arrive at a constant result. Again, Appellants conclude that the Examiner has erred in rejecting the claim language as anticipated.

**II. Rejection under 35 USC §103 as unpatentable over  
Linden in view of Shaylor**

**Claim 5**

Claim 5 recites a program optimization method for translating, into machine code, the source code for a program written in an object-oriented programming language, and for optimizing the program comprising the steps of detecting one command dynamically during execution, of the commands in the program, for which a method call destination can be identified, and for which the processing speed can be increased by identifying the method call destination, and dynamically generating a path wherefor the method call destination for the detected command is limited in order to increase the processing speed of the command.



**Serial No. 10/085,455**

The Examiner has rejected Claim 5 as unpatentable over the teachings of Linden in view of Shaylor. Appellants rely on the discussion and arguments presented above with regard to the teachings of the Linden patent publication. Applicants further note that the Shaylor reference does not supply those teachings which are missing from Linden.

The Examiner cites Col. 2, lines 5-56 of Shaylor, and lists the terms "method call" and "inlining". The cited passage from Col. 2 primarily describes the disadvantages of just in time (so-called "JIT") compiling. The description in lines 48-56 of Col. 2 teaches that the overhead of method calls can be reduced by "inlining" which copies the code of a target method into a calling method. Copying the code of the target method into a calling method is not the same as or suggestive of detecting a command for which a method call destination can be identified and for which the processing speed can be increased by identifying the method call destination. Further, the cited teachings from Col. 4, lines 5-42 state that Shaylor addresses program optimization by "heavy use of inlining techniques when the code involves target method calls". That passage also does not teach or suggest detecting a command for which a method call destination can be identified and for which the processing

**Serial No. 10/085,455**

speed can be increased by identifying the method call destination. Finally, the Examiner cites Col. 6, lines 50-62 against the claimed step of dynamically generating a path wherefor the method call destination for the detected command is limited in order to increase processing speed. The cited passage teaches that either inlining or direct method calls can be implemented. The passage further states that inlining causes code size to increase, which effectively teaches away from the claimed steps for increasing processing speed for a command. The alternative direct method call provided in the Shaylor passage is not the same as or suggestive of dynamically generating a path where a call destination is limited to increase processing speed.

For a determination of obviousness, the prior art must teach or suggest all of the claim limitations. "All words in a claim must be considered in judging the patentability of that claim against the prior art" (In re Wilson, 424 F. 2d 1382, 1385, 165 USPQ 494, 496 (C.C.P.A. 1970)). Since the cited references fail to teach each and every one of the claim limitations, a *prima facie* case of obviousness has not been established by the Examiner.

Serial No. 10/085,455

**CONCLUSION**

Appellants respectfully assert that the Examiner has erred in rejecting Claims 1-4 and 6-16 as anticipated by the Linden patent publication and has erred in rejecting Claim 5 as unpatentable over Linden in view of Shaylor.

Based on the foregoing arguments, Appellants request that the decisions of the Examiner be overturned by the Board and that the claims be passed to issuance.

Respectfully submitted,  
M. Kawahito, et al

By: Anne Vachon Dougherty  
Anne Vachon Dougherty  
Registration No. 30,374  
Tel. (914) 962-5910

APPENDIX OF CLAIMS

1. A program optimization method for translating, into machine code, source code for a program written in a programming language, and for dynamically optimizing said program comprising the steps of:

performing a dynamic analysis during execution to determine whether the execution speed of said program can be increased by fixing, in a specific state, a parameter for a predetermined command in said program; and

employing results of said analysis for the dynamic generation, in said program, of a path along which said parameter of said predetermined command is fixed in said specific state.

2. The program optimization method according to claim 1, wherein said step of generating a path includes the steps of:

executing said program and obtaining statistical data for the appearance frequency of each available state wherein, according to said results of said analysis, said parameter of said predetermined command may be set; and

**Serial No. 10/085,455**

employing said obtained statistical data to dynamically generate said path.

3. A program optimization method for translating, into machine code, the source code for a program written in a programming language, and for optimizing said program comprising the steps of:

executing a program to obtain statistical data for an appearance frequency of each available state in which a parameter of a predetermined command in said program may be set; and

employing said obtained statistical data to dynamically generate a machine language program that includes, as the compiling results, a path along which said parameter of said predetermined command is fixed in a specific state.

4. The program optimization method according to claim 3, further comprising a step of:

generating a machine language program that does not

**Serial No. 10/085,455**

include, as a compiling result, a path along which said parameter of said predetermined command is fixed in a specific state.

5. A program optimization method for translating, into machine code, the source code for a program written in an object-oriented programming language, and for optimizing said program comprising the steps of:

detecting one command dynamically during execution, of the commands in said program, for which a method call destination can be identified, and for which the processing speed can be increased by identifying said method call destination; and

dynamically generating a path wherefor said method call destination for said detected command is limited in order to increase the processing speed of said command.

6. A program optimization method for translating into machine code, the source code for a program written in a programming language, and for optimizing said program comprising the steps of:

**Serial No. 10/085,455**

detecting dynamically during program execution one command, of the commands in said program, for which a variable can be limited to a predetermined constant value, and for which the processing speed can be increased by limiting said variable to said constant value; and

generating a path along which said constant value for said variable of said detected command is fixed.

7. A dynamic compiler for translating into machine code the source code for a program written in a programming language, and for optimizing the resultant program comprising:

an impact analysis unit for performing an analysis to dynamically determine during execution how much the execution speed of said program can be increased by fixing, in a specific state, a parameter of a predetermined command in said program; and

a specialization unit for employing the analysis results obtained by said impact analysis unit to generate, in said program, a specialized path along which said parameter of said predetermined command is fixed in said specific state.

**Serial No. 10/085,455**

8. The compiler according to claim 7, further comprising:

a data specialization selector for, when said program is executed, obtaining statistical data for the appearance frequency of each state obtained by said impact analysis unit, and for determining the state in which said parameter of said predetermined command is to be set, wherein said specialization unit generates a specialized path along which said parameter of said predetermined command is fixed in a state determined by said data specialization selector.

9. The compiler according to claim 8, wherein, in accordance with the state of said program at execution, said specialization unit generates, in said program, a branching process for selectively performing a specialized path and an unspecialized path; and wherein, while taking into account a delay due to the insertion of said branching process, said data specialization selector determines a state in which said parameter of said predetermined command is fixed.

10. A computer comprising:

an input device for receiving source code for a program;



**Serial No. 10/085,455**

a dynamic compiler for translating said source code to compile said program and for converting said compiled program into machine language code; and

a processor for executing said machine language code,  
wherein said dynamic compiler includes

means for performing a dynamic analysis to determine during execution whether the execution speed of said program can be improved by fixing in a specific state a parameter of a predetermined command in said program, and

means for generating in said program, based on the analysis results, a path along which said parameter of said predetermined command is fixed in said specific state and for compiling said program, and

wherein said compiler outputs, as the compiled results, said machine language code that includes said path along which the state of said parameter is fixed.

11. A computer comprising:

an input device, for receiving source code for a program;

**Serial No. 10/085,455**

a dynamic compiler, for translating said source code to compile said program and for converting said compiled program into machine language code; and

a processor, for executing said machine language code, wherein said dynamic compiler includes

means for obtaining statistical data for the appearance frequency of each available state wherein a parameter for a predetermined command in said program may be set when said program is executed, and for employing said statistical data to determine a state in which said parameter of said predetermined command is to be fixed, and

means for generating a specialized path along which said parameter of said predetermined command is fixed in said determined state, and for compiling said program, and

wherein said compiler outputs, as the compiled results, said program as said machine language code that includes said specialized path.

12. The computer according to claim 11, comprising:

said compiler further includes

means for compiling said program without

**Serial No. 10/085,455**

generating a specialized path,

wherein, when said state of said parameter to be fixed can not be determined, said means for determining the state of said parameter of said predetermined command outputs, as compiled results, said program in said machine language code, which is generated by said means for compiling said program without generating said specialized path, that does not include said specialized path.

13. A support program, for controlling a computer to support generation of a program, which permits said computer to perform:

a function for performing a dynamic analysis to determine during execution whether the execution speed of said program can be increased by fixing a parameter of a predetermined command of said computer program in a specific state; and

a function for generating in said program, based on the analysis results, a path along which said parameter of said predetermined command is fixed in said specific state.

**Serial No. 10/085,455**

14. A support program, for controlling a computer to support generation of a program, which permits said computer to perform:

    a function for executing said program and obtaining statistical data for the appearance frequency of each available state wherein said parameter of said predetermined command of said program may be set; and

    a function for generating in said program, based on said statistical data, a path along which said parameter of said predetermined command is fixed in said specific state.

15. A storage medium on which input means of a computer stores a computer-readable support program, for controlling said computer to support generation of a program, that permits said computer to perform:

    a function for performing a dynamic analysis to determine during execution whether the execution speed of said program can be increased by fixing a parameter of a predetermined command of said computer program in a specific state; and

    a function for generating in said program, based on the analysis results, a path along which said parameter of said predetermined command is fixed in said specific state.

**Serial No. 10/085,455**

16. A storage medium on which input means of a computer stores a computer-readable support program, for controlling said computer to support generation of a program, that permits said computer to perform:

    a function for executing said program and obtaining statistical data for the appearance frequency of each available state wherein said parameter of said predetermined command of said program may be set; and

    a function for generating in said program, based on said statistical data, a path along which said parameter of said predetermined command is fixed in said specific state.

**Serial No. 10/085,455**

**EVIDENCE APPENDIX**

There is no additional evidence.

Serial No. 10/085,455

**RELATED PROCEEDINGS APPENDIX**

There are no related proceedings.